REMARKS

Reconsideration of this application as amended is respectfully requested.

In the Office Action dated July 27, 2009, claims 1-29 are pending. Claims 1-29 stand rejected. In this response, claims 1-3, 6, 8-10, 13-15, 17, 23 and 29 have been amended. No new claims have been added. No claims have been canceled. Thus, claims 1-29 remain pending. Support for the amendments can be found throughout the specifications as filed. No new matter has been added.

Rejections under 35 U.S.C. § 102(b)

Claims **1-2 and 15-16**

Claims 1-2 and 15-16 stand rejected under 35 U.S.C. §102(b) as being anticipated by "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (hereinafter "Luk"). However, applicants respectfully submit that applicants' claims 1-2 and 15-16, as amended, are not anticipated by the cited reference.

Specifically, for example, independent claim 1 includes the limitations:

> "analyzing source codes of a main thread having one or more delinquent loads, the one or more delinquent loads representing loads which likely suffer cache misses during an execution of the main thread, the source codes including one or more code regions, each code region corresponding to a sequence of instructions representing an iteration loop in the source codes, the one or more code regions sharing at least one instruction in the source codes;
> estimating a communication cost between the main thread and each code region;
> selecting a code region from the one or more code regions for one or more helper threads with respect to the main thread based on the communication cost; and
> generating codes for the one or more helper threads, the one or more helper threads being speculatively executed in parallel with the main thread to perform one or more tasks for the selected code region of the main thread"
> (emphasis added)

Applicants submit that Luk fails to teach or suggest the limitation of selecting a code region from one or more code regions representing iteration loops sharing instructions in source codes of a main thread based on an estimation of communication cost between the code region and the main thread to generate one or more helper threads.

Rather, Luk provides software-controlled pre-execution schemes to help programmers writing memory-friendly programs using pre-execution threads and for compiler to insert pre-execution (Luk, page 40, col. 2, par. 4 – page 41, col. 1, par. 1). Nowhere does Luk teach or suggest selecting a code region from one or more code regions representing iteration loops sharing instructions in source codes of a main thread based on an estimation of communication cost between the code region and the main thread to generate one or more helper thread.

The Office Action states that "Luk … disclosing … selecting a code region (Fig. 2(a) "arcin=(arc_ *) … while (arcin) {…}") from one or more code regions representing iteration loops (Fig. 2(a) "for(;i<trips;){…}") (Office Action , page 2). It appears that the Office Action alleges Luk discloses selecting an iteration loop from one or more iteration loops in a source code. Applicants respectfully disagree with the Office Action.

Instead, Luk describes inserting four lines of pre-execution code for a code region corresponding to instructions comprising only a portion of a for-loop and a while loop in a source code (Luk, Figure 2(b), page 42). Here, instructions for an incomplete loop cannot represent instructions for an iteration loop. Therefore, Luk's code region does not correspond to instructions represent an iteration loop. It is respectfully submitted that that one with ordinary skill in the art would not recognize selecting an iteration loop from one or more code iteration loops in a source code based on the teachings of Luk.

In order to anticipate a claim, each and every limitation of the claim must be taught by the cited reference. It is respectfully submitted that Luk fails to disclose the limitations set forth above. Therefore, it is respectfully submitted that independent claim 1, as amended, is not anticipated by Luk.

Independent claim15, as amended, include similar limitations as noted above. Therefore, for at least the similar reasons as discussed above, it is respectfully submitted that claim 15, as amended, are not anticipated by Luk.

Given that claims 2 and 16, as amended, depend from and include all limitations of one of independent claims 1 and 15, as amended, applicants respectfully submit that claims 2, and 16, as amended, are not anticipated by Luk.

Claims **8-9**

Claims 8-9 stand rejected under 35 U.S.C. §102(b) as being anticipated by Luk. However, applicants respectfully submit that applicants' claims 8-9, as amended, are not anticipated by the cited reference.

Specifically, for example, independent claim 8 includes the limitations:

> "generating software codes for the one or more helper threads, the one or more helper threads to be speculatively executed in parallel with the main thread to perform one or more prefetching tasks for the selected code region of the main thread,
> wherein the generated software codes include synchronization codes for the one or more helper threads to synchronize with the main thread during the execution"
> (emphasis added)

Applicants submit that Luk fails to teach or suggest the limitation of generating software codes including synchronization codes for a helper thread to synchronize with a main thread during execution.

Rather, Luk teaches extending an instruction set with three new instructions to allow a main thread to spawn and terminate a pre-execution thread (Luk, sec. 3.1, page 44). Luk specifically states that only a main thread, i.e. not a pre-execution thread, is allowed to spawn and terminate a pre-execution thread (Luk, sec 3.1, page 44). Luk also describes hardware based heuristics for a hardware to terminate a pre-execution thread. (Luk, sec 3.5, page 46). Thus, Luk's pre-execution threads may be controlled by a main thread or a hardware mechanism. However, Luk is completely silent about generating software codes including synchronization codes for a helper thread to synchronize with a main thread during execution.

In order to anticipate a claim, each and every limitation of the claim must be taught by the cited reference. It is respectfully submitted that Luk fails to disclose the limitations set forth above. Therefore, it is respectfully submitted that independent claim 8 and dependent claim 9, as amended, are not anticipated by Luk.

Rejections Under 35 U.S.C. 103(a)

Claims **3-4, 10-11, 5-7, 12-14, 17-19, 22-25, 28-29, 20-21, 26-27**

Claims 3-4 and 10-11 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Luk in view of "Exploiting hardware Performance Counters with Flow and Context Sensitive Profiling" by Ammons et al. (hereinafter "Ammons"). Claims 5-7, 2(12)-14, 17-19, 22-25 and 28-29 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Luk in view of "Data Prefetching by Dependence Graph Precomputation" by Annavaram et al. (hereinafter "Annavaram"). Claims 20-21 and 26-27 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Luk in view of Annavaram and US Patent No. 7,243,267 to Klemm et al. (hereinafter "Klemm"). Applicants hereby reserve the right to swear behind Klemm at a later date. However, applicants respectfully submit that applicants' claims 3-4, 10-11, 5-7, 12-14, 17-19, 22-25, 28-29, 20-21 and 26-27, as amended, are patentable over the cited references.

These claims depend from one of the above independent claims. It is respectfully submitted that none of the cited references herein, individually or in combination, disclose or suggest the limitations set forth above.

Further, the Office Action notes that "Regarding Claim 7: … Luk discloses … determining a synchronization period for the helper thread to synchronize the main thread with the helper thread … (pg. 46, col. 1, 2nd par. A pre-execution thread must be terminated if its next PC is out of the acceptable range" (Office Action, page 10). It appears that the Office Action alleges Luk's acceptable address range is a synchronization period for a thread to synchronize with a main thread within the synchronization period. Applicants respectfully disagree.

Instead, Luk terminates a pre-execution thread if its next PC is out of an acceptable range imposed by an operating system to preserve correctness (Luk, pg. 46, col. 1, 2nd par). Here, Luk merely exhibits the common-sense approach that a thread should be terminated if the PC for the next instruction falls outside the acceptable range of executable addresses of the operating system. Whether the PC of a thread falls outside the acceptable address range imposed by an operating system has nothing to do with whether the thread synchronizes with a main thread. Thus, Luk does terminate a thread to synchronize with a main thread when the

PC of the thread is out of an acceptable range. It is respectfully submitted that one with ordinary skill in the art would not recognize determining a synchronization period for a helper thread to synchronize a main thread with the helper thread within the synchronization period based on Luk's teaching.

Furthermore, there are neither motivations nor suggestions to combine the cited references. Therefore, for the reasons similar to those discussed above, these claims are patentable over the cited references.

CONCLUSION

In view of the foregoing, it is respectfully submitted that the applicable rejections and objections have been overcome.

Authorization is now given to charge our Deposit Account No. 02-2666 for any charges that may be due. Furthermore, if an extension is required, then applicant hereby requests such an extension.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP

Date: October 23, 2009

/Vincent WenJeng Lue, Reg. No. 56,564/

Vincent_Lue@bstz.com
Vincent WenJeng Lue
Reg. No. 56,564

1279 Oakmead Parkway
Sunnyvale, California 94086-4040
(408) 720-8300